Offline Reinforcement Learning Part 2

CS 285

Instructor: Sergey Levine
UC Berkeley
Offline Reinforcement Learning

Formally:

\[ D = \{(s_i, a_i, s'_i, r_i)\} \]

\[ s \sim d^{\pi_\beta}(s) \]

\[ a \sim \pi_\beta(a|s) \]

\[ s' \sim p(s'|s, a) \]

\[ r \leftarrow r(s, a) \]

RL objective:

\[ \max_{\pi} \sum_{t=0}^{T} E_{s_t \sim d(s), a_t \sim \pi(a|s)} [\gamma^t r(s_t, a_t)] \]
Where do we suffer from distribution shift?

\[ Q(s, a) \leftarrow r(s, a) + \max_{a'} Q(s', a') \]

\[ Q(s, a) \leftarrow r(s, a) + E_{a' \sim \pi_{\text{new}}} [Q(s', a')] \]

\[ y(s, a) \]

what is the objective?

\[ \min_{Q} E_{(s, a) \sim \pi_{\beta}(s, a)} [(Q(s, a) - y(s, a))^2] \]

behavior policy

target value

expect good accuracy when \( \pi_{\beta}(a|s) = \pi_{\text{new}}(a|s) \)

even worse: \( \pi_{\text{new}} = \arg \max_{\pi} E_{a \sim \pi(a|s)} [Q(s, a)] \)

(what if we pick \( x^* \leftarrow \arg \max_{x} f_\theta(x) \)?)

how often does that happen?

how well it does

how well it \textit{thinks} it does (Q-values)

Kumar, Fu, Tucker, Levine. \textit{Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction}. NeurIPS ’19
How do prior methods address this?

\[ Q(s, a) \leftarrow r(s, a) + E_{a' \sim \pi_{\text{new}}}[Q(s', a')] \]

\[ \pi_{\text{new}}(a|s) = \arg \max_{\pi} E_{a \sim \pi(a|s)}[Q(s, a)] \quad \text{s.t.} \quad D_{KL}(\pi||\pi_\beta) \leq \epsilon \]

This solves distribution shift, right?

No more erroneous values?

**Issue 1:** we usually don’t know the behavior policy \( \pi_\beta(a|s) \)

- human-provided data
- data from hand-designed controller
- data from many past RL runs
- all of the above

**Issue 2:** this is both *too pessimistic* and *not pessimistic enough*

“policy constraint” method

**very** old idea (but it had no single name?)

Todorov et al. [passive dynamics in linearly-solvable MDPs]

Kappen et al. [KL-divergence control, etc.]

trust regions, covariant policy gradients, natural policy gradients, etc.

used in some form in recent papers:

Fox et al. ‘15 (“Taming the Noise…”)

Fujimoto et al. ‘18 (“Off Policy…”)

Jaques et al. ‘19 (“Way Off Policy…”)

Kumar et al. ‘19 (“Stabilizing…”)

Wu et al. ‘19 (“Behavior Regularized…”)
Explicit policy constraint methods

What kinds of constraints can we use?

KL-divergence: $D_{KL}(\pi \| \pi_\beta)$

+ easy to implement (more on this later)
- not necessarily what we want

support constraint: $\pi(a|s) \geq 0$ only if $\pi_\beta(a|s) \geq \epsilon$

+ can approximate with MMD
- significantly more complex to implement
+ much closer to what we really want

For more information, see:


Kumar, Fu, Tucker, Levine. *Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction*. ‘19

Explicit policy constraint methods

How do we implement constraints?

1. Modify the actor objective

$$\theta \leftarrow \arg \max_{\theta} E_{s \sim D} \left[ E_{a \sim \pi_\theta(s)} [Q(s, a)] \right]$$

$$\theta \leftarrow \arg \max_{\theta} E_{s \sim D} \left[ E_{a \sim \pi_\theta(s)} [Q(s, a) + \lambda \log \pi_\beta(a|s)] + \lambda \mathcal{H}(\pi(a|s)) \right]$$

$$D_{KL}(\pi \| \pi_\beta) = E_\pi [\log \pi(a|s) - \log \pi_\beta(a|s)] = -E_\pi [\log \pi_\beta(a|s)] - \mathcal{H}(\pi)$$

2. Modify the reward function

$$\tilde{r}(s, a) = r(s, a) - D(\pi, \pi_\beta)$$

simple modification to directly penalize divergence
also accounts for future divergence


generally, the best modern offline RL methods do not do either of these things
Implicit policy constraint methods

\[ \pi_{\text{new}}(a|s) = \arg \max_{\pi} E_{a \sim \pi(a|s)}[Q(s, a)] \ \text{s.t.} \ D_{\text{KL}}(\pi \| \pi_\beta) \leq \epsilon \]

\[ \pi^*(a|s) = \frac{1}{Z(s)} \pi_\beta(a|s) \exp \left( \frac{1}{\lambda} A^\pi(s, a) \right) \]

straightforward to show via duality

approximate via **weighted** max likelihood!

\[ \pi_{\text{new}}(a|s) = \arg \max_{\pi} E_{(s,a) \sim \pi_\beta} \left[ \log \pi(a|s) \frac{1}{Z(s)} \exp \left( \frac{1}{\lambda} A^{\pi_{\text{old}}}(s, a) \right) \right] \]

samples from dataset
\[ a \sim \pi_\beta(a|s) \]

critic can be used to give us this

See also:
Peters et al. (REPS)
Rawlik et al. (“psi-learning”)
...many follow-ups

Peng*, Kumar*, Levine. **Advantage-Weighted Regression.** ‘19

Nair, Dalal, Gupta, Levine. **Accelerating Online Reinforcement Learning with Offline Datasets.** ‘20
Implicit policy constraint methods

$$L_C(\phi) = E_{(s,a,s') \sim D} \left[ (Q_\phi(s, a) - (r(s, a) + \gamma E_{a' \sim \pi_\theta(a'|s')} [Q_\phi(s', a')])^2 \right]$$

$$L_A(\theta) = -E_{(s,a) \sim \pi_\beta} \left[ \log \pi_\theta(a|s) \frac{1}{Z(s)} \exp \left( \frac{1}{\lambda} A_{\pi_{\text{old}}}(s, a) \right) \right]$$

1. $\phi \leftarrow \phi - \alpha \nabla_\phi L_C(\phi)$
2. $\theta \leftarrow \theta - \alpha \nabla_\theta L_A(\theta)$

$$Q(s, a) \leftarrow r(s, a) + E_{a' \sim \pi_{\text{new}}} [Q(s', a')]$$

$$\pi_{\text{new}}(a|s) = \arg \max_{\pi} E_{a \sim \pi(a|s)} [Q(s, a)] \text{ s.t. } D_{\text{KL}}(\pi || \pi_\beta) \leq \epsilon$$

Peng*, Kumar*, Levine. Advantage-Weighted Regression. ’19
Nair, Dalal, Gupta, Levine. Accelerating Online Reinforcement Learning with Offline Datasets. ’20
Can we also avoid all OOD actions in the Q update?

\[
Q(s, a) \leftarrow r(s, a) + E_{a' \sim \pi_{\text{new}}} [Q(s', a')]
\]

\[
V(s') \quad \text{just another neural network}
\]

\[
V \leftarrow \arg \min_V \frac{1}{N} \sum_{i=1}^{N} \ell(V(s_i), Q(s_i, a_i))
\]

e.g., MSE loss \((V(s_i) - Q(s_i, a_i))^2\) this action comes from \(\pi_\beta\)

not from \(\pi_{\text{new}}\)

expectile: \(\ell^\tau_2(x) = \begin{cases} 
(1 - \tau)x^2 & \text{if } x > 0 \\
\tau x^2 & \text{else}
\end{cases}\)

\[
V(s) \leftarrow \max_{a \in \Omega(s)} Q(s, a)
\]

\[
\Omega(s) = \{a : \pi_\beta(a|s) \geq \epsilon\}
\]

if we use \(\ell^\tau_2\) for big \(\tau\)
Implicit Q-learning (IQL)

Q-learning with *implicit* policy improvement

\[
Q(s, a) \leftarrow r(s, a) + V(s') \\
V \leftarrow \arg\min_V \frac{1}{N} \sum_{i=1}^{N} \ell_2^\tau(V(s_i), Q(s_i, a_i))
\]

\[
V(s) \leftarrow \max_{a \in \Omega(s)} Q(s, a)
\]

\[
\Omega(s) = \{a : \pi_\beta(a|s) \geq \epsilon\}
\]

if we use \(\ell_2^\tau\) for big \(\tau\)

\[
Q(s, a) \leftarrow r(s, a) + \max_{a' \in \Omega(s')} Q(s', a')
\]

“implicit” policy

\[
\pi_{\text{new}}(a|s) = \delta(a = \arg \max_{a \in \Omega(s)} Q(s, a))
\]

Now we can do value function updates without ever risking out-of-distribution actions!

We’ll see results soon, but first let’s talk about Option 2…
Conservative Q-Learning
Conservative Q-learning (CQL)

how well it does
how well it thinks it does (Q-values)

\[ \hat{Q}^\pi = \arg \min_{Q} \max_{\mu} \alpha E_{s \sim D, a \sim \mu(s)}[Q(s, a)] \]

regular objective

\[ + E_{(s, a, s') \sim D} \left[ (Q(s, a) - (r(s, a) + E_{\pi}[Q(s', a')]))^2 \right] \]

term to push down big Q-values

can show that \( \hat{Q}^\pi \leq Q^\pi \) for large enough \( \alpha \)

true Q-function
Conservative Q-learning (CQL)

A better bound: \( \hat{Q}_\pi = \arg \min_Q \max_\mu \alpha E_{s \sim D, a \sim \mu(a | s)}[Q(s, a)] - \alpha E_{(s, a) \sim D}[Q(s, a)] \\
+ E_{(s, a, s') \sim D} \left[ (Q(s, a) - (r(s, a) + E_{\pi}[Q(s', a')]))^2 \right] \)

no longer guaranteed that \( \hat{Q}_\pi(s, a) \leq Q_\pi(s, a) \) for all \((s, a)\)

but guaranteed that \( E_{\pi(a | s)}[\hat{Q}_\pi(s, a)] \leq E_{\pi(a | s)}[Q_\pi(s, a)] \) for all \(s \in D\)

Conservative Q-learning (CQL)

1. Update $\hat{Q}^\pi$ w.r.t. $\mathcal{L}_{\text{CQL}}(\hat{Q}^\pi)$ using $\mathcal{D}$

2. Update policy $\pi$

   if actions are discrete:

   $$\pi(a|s) = \begin{cases} 
   1 & \text{if } a = \text{arg max}_a \hat{Q}(s, a) \\
   0 & \text{otherwise}
   \end{cases}$$

   if actions are continuous:

   $$\theta \leftarrow \theta + \alpha \nabla_{\theta} \sum_i E_{a \sim \pi_\theta(a|s_i)} [\hat{Q}(s_i, a)]$$
Conservative Q-learning (CQL)

\[ \hat{Q}^\pi = \arg \min_Q \max_\mu \alpha E_{s \sim D, a \sim \mu(a|s)}[Q(s, a)] - \alpha E_{(s, a) \sim D}[Q(s, a)] - R(\mu) \\
+ E_{(s, a, s') \sim D} \left[ (Q(s, a) - (r(s, a) + E_\pi[Q(s', a')]))^2 \right] \]

\[ \mathcal{L}_{CQL}(\hat{Q}^\pi) \]

common choice: \[ R = E_{s \sim D}[\mathcal{H}(\mu(\cdot|s))] \] maximum entropy regularization

optimal choice: \[ \mu(a|s) \propto \exp(Q(s, a)) \]

\[ E_{a \sim \mu(a|s)}[Q(s, a)] = \log \sum_a \exp(Q(s, a)) \]

for discrete actions: just calculate directly

for continuous actions: use importance sampling to estimate \[ E_{a \sim \mu(a|s)}[Q(s, a)] \]
Model-Based Offline RL
How does **model-based** RL work?

\[
\hat{p}(s_{t+1}|s_t, a_t) \quad \pi(a_t|s_t)
\]

what goes wrong when we can’t collect more data?

...so the model’s predictions are invalid these states are OOD

the model answers “what if” questions
MOPO: Model-Based Offline Policy Optimization

solution: “punish” the policy for exploiting

\[ \tilde{r}(s, a) = r(s, a) - \lambda u(s, a) \]

...and then use any existing model-based RL algorithm

Yu*, Thomas*, Yu, Ermon, Zou, Levine, Finn, Ma. MOPO: Model-Based Offline Policy Optimization. ‘20

See also: Kidambi et al., MOREL: Model-Based Offline Reinforcement Learning. ’20 (concurrent)
MOPO: Theoretical Analysis

\[ \tilde{r}(s, a) = r(s, a) - \lambda u(s, a) \]

we can represent the value function

model error is bounded (above) by \( u(s, a) \)

**Theorem 4.4.** Under Assumption 4.2 and 4.3, the learned policy \( \hat{\pi} \) in MOPO (Algorithm 1) satisfies

true return of policy trained under model

\[ \eta_M(\hat{\pi}) \geq \sup_\pi \{ \eta_M(\pi) - 2\lambda \epsilon_u(\pi) \} \]  \hspace{1cm} \text{(11)}

\[ \epsilon_u(\pi) := \mathbb{E}_{(s,a) \sim p_T} [u(s,a)] \]

some implications:

\[ \eta_M(\hat{\pi}) \geq \eta_M(\pi^B) - 2\lambda \epsilon_u(\pi^B) \]

➢ improves over behavior policy

\[ \eta_M(\hat{\pi}) \geq \eta_M(\pi^*) - 2\lambda \epsilon_u(\pi^*) \]

➢ quantifies “optimality gap” in terms of model error

Yu*, Thomas*, Yu, Ermon, Zou, Levine, Finn, Ma. MOPO: Model-Based Offline Policy Optimization. ‘20
COMBO: Conservative Model-Based RL

**Basic idea:** just like CQL minimizes Q-value of policy actions, we can minimize Q-value of model state-action tuples

\[
\hat{Q}^{k+1} \leftarrow \arg \min_{\hat{Q}} \beta \left( \mathbb{E}_{s,a \sim \rho(s,a)} [Q(s,a)] - \mathbb{E}_{s,a \sim \mathcal{D}} [Q(s,a)] \right) + \frac{1}{2} \mathbb{E}_{s,a,s' \sim d_f} \left[ (Q(s,a) - \hat{Q}(s,a)) \right]^2.
\] (4)

**Intuition:** if the model produces something that looks clearly different from real data, it’s easy for the Q-function to make it look bad

Yu, Kumar, Rafailov, Rajeswaran, Levine, Finn. **COMBO: Conservative Offline Model-Based Policy Optimization.** 2021.
Trajectory Transformer

Basic ideas:

1. train a joint state-action model:
   \[ p_\beta(\tau) = p_\beta(s_1, a_2, \ldots, s_T, a_T) \]

2. use a big expressive model (a Transformer)

The model:

<table>
<thead>
<tr>
<th>s_{1,2}</th>
<th>s_{1,3}</th>
<th>a_{1,1}</th>
<th>a_{1,2}</th>
<th>s_{2,1}</th>
<th>a_{T,d_a}</th>
</tr>
</thead>
</table>

Why does this work?

- generating high-probability trajectories avoids out-of-distribution states & actions
- using a really big model works well in offline mode (lots of compute, captures complex behavior policies)

How to do control:

- beam search, but use \( \sum_t r(s_t, a_t) \) instead of probability
- 1. given current sequence, sample next tokens from model
- 2. store top \( K \) tokens with highest cumulative reward
- 3. move on to next token

Summary, Applications, Open Questions
Which offline RL algorithm do I use?

If you want to *only* train offline...

- **Conservative Q-learning**
  - + just one hyperparameter
  - + well understood and widely tested
- **Implicit Q-learning**
  - + more flexible (offline + online)
  - - more hyperparameters

If you want to *only* train offline and finetune online

- **Advantage-weighted actor-critic (AWAC)**
  - + widely used and well tested
- **Implicit Q-learning**
  - + seems to perform much better!

If you have a good way to train models in your domain

- **COMBO**
  - + similar properties as CQL, but benefits from models
  - - not always easy to train a good model in your domain!
- **Trajectory transformer**
  - + very powerful and effective models
  - - extremely computationally expensive to train and evaluate
The power of offline RL

**standard real-world RL process**

1. instrument the task so that we can run RL
   - safety mechanisms
   - autonomous collection
   - rewards, resets, etc.

2. wait a long time for online RL to run

3. change the algorithm in some small way

4. throw it all in the garbage and start over for the next task

**offline RL process**

1. collect initial dataset
   - human-provided
   - scripted controller
   - baseline policy
   - all of the above

2. Train a policy offline

3. change the algorithm in some small way

4. collect more data, add to growing dataset

5. keep the dataset and use it again for the next project!
Offline RL in robotic manipulation: MT-Opt, AMs

Kalashnikov, Irpan, Pastor, Ibarz, Herzong, Jang, Quillen, Holly, Kalakrishnan, Vanhoucke, Levine. QT-Opt: Scalable Deep Reinforcement Learning of Vision-Based Robotic Manipulation Skills


➢ 12 different tasks
➢ Thousands of objects
➢ Months of data collection

New hypothesis: could we learn these tasks without rewards using goal-conditioned RL?


reuse the same exact data
Actionable Models: Offline RL with Goals

- No reward function at all, task is defined entirely using a goal image!
- Uses a conservative offline RL method designed for goal-reaching, based on CQL
- Works very well as an unsupervised pretraining objective!

1. Train goal-conditioned Q-function with offline RL
2. Finetune with a task reward and limited data
More examples

Early 2020: Greg Kahn collects 40 hours of robot navigation data


Late 2020: Dhruv Shah uses it to build goal-conditioned navigation system


Early 2021: Dhruv Shah uses the same dataset to train an exploration system

Takeaways, conclusions, future directions

"the gap"

1. Collect a dataset using any policy or mixture of policies
2. Run offline RL on this dataset to learn a policy
3. Deploy the policy in the real world

"the dream"

- An offline RL workflow
  - Supervised learning workflow: train/test split
  - Offline RL workflow: ???
- Statistical guarantees
  - Biggest challenge: distributional shift/counterfactuals
  - Can we make any guarantees?
- Scalable methods, large-scale applications
  - Dialogue systems
  - Data-driven navigation and driving

A starting point: Kumar, Singh, Tian, Finn, Levine. A Workflow for Offline Model-Free Robotic Reinforcement Learning. CoRL 2021